



Let's learn GenerativeAI with DTsense!

Generative AI Academy

Extra Module: Streamlit

Introduction to Streamlit

Building Frontend for AI/Data Applications

I Nyoman Prayana Trisna, M.Cs.

Alvin Rindra Fazrie, M.Sc.

Data App with Streamlit

Pengenalan Streamlit

Membuat kode ataupun program untuk model *machine learning* saja tidak cukup, karena orang biasa tidak akan bisa menggunakannya. Di sini, penting untuk menerapkan model sedemikian rupa sehingga siapa pun dapat menggunakannya dengan mudah. Selain menggambarkan model, penting juga untuk menggambarkan visual dengan menarik dari data yang diberikan. Ada opsi berbeda seperti Django dan Flask dengan Python untuk menerapkan model maupun untuk memvisualisasikan data. Saat menggunakan *framework* ini, kita juga perlu memiliki pengetahuan tentang membangun *front end* menggunakan HTML, CSS, dan JS.

Streamlit adalah *open-source framework* di Python yang memungkinkan untuk menerapkan tidak hanya model machine learning tetapi juga aplikasi Python apa pun dalam waktu yang singkat. Ini juga memungkinkan untuk menggambarkan dan memvisualisasikan data dengan lebih interaktif. Proses pembuatannya yang sederhana menjadi keunggulan untuk Streamlit dalam membangun aplikasi visualisasi ataupun *machine learning*.



Streamlit

Keunggulan Streamlit

Dibandingkan dengan framework Python lainnya, dalam pengembangan aplikasi visualisasi ataupun machine learning, Streamlit memiliki keunggulan:

- Sederhana
Streamlit jauh lebih mudah dan sederhana digunakan oleh *data scientist* untuk membangun *dashboard*
- Interaktif
Visualisasi dalam Streamlit dapat diinterasikan dengan menarik. Menggunakan Plotly, visual yang dibangun menjadi lebih menarik
- Mudah diintegrasikan
Streamlit dapat diintegrasikan dengan berbagai library dan framework populer untuk *data science* seperti Pandas, Matplotlib, Plotly, dan Scikit-learn
- *Deployment* yang cepat
Setelah aplikasi dibuat, Streamlit juga menyediakan cara yang mudah untuk me-*deploy* aplikasi ke berbagai platform.

Instalasi Streamlit

Streamlit dapat di-*install* secara mudah melalui *pip*.

```
>pip install streamlit
```

Biasanya, dalam membuat suatu aplikasi dalam Streamlit, kita membutuhkan *virtual environment* terlebih dahulu. Python Virtual Environment atau sering disebut juga dengan Python *virtualenv* adalah alat yang membantu programmer membuat dan mengelola *environment* terpisah untuk setiap proyek Python. Cara membuat virtual environment sederhana dalam Python ditunjukkan pada gambar di bawah.

```
>python -m venv <nama_virtual_env>
```

Cara menggunakan *virtual environment* yang sudah dibuat adalah seperti gambar di bawah. Perlu dicatat bahwa *<nama_virtual_env>* disesuaikan dengan nama *virtual environment* yang diberikan.

```
><nama_virtual_env>\Scripts\activate
```

Setelah me-run *snippet* di atas, barulah kita dapat me-*install* Streamlit serta library lainnya yang dibutuhkan.

Untuk menjalankan aplikasi Streamlit, mula-mula buat kode aplikasi sederhana terlebih dahulu. Sebagai contoh, kita buat program *hello world* sederhana seperti di bawah. Kode ini disimpan dengan nama *hello.py*, misalnya.

```
1 import streamlit as st  
2 st.title("hello world")
```

Bila sudah, kita bisa menjalankan aplikasi sederhana ini dengan *command* seperti di bawah ini.

```
>streamlit run hello.py
```

Maka *browser* akan langsung terbuka dan menjalankan aplikasi yang telah dibuat dalam *hello.py*.

Deploy ⋮

☞ hello world

Fitur dalam Streamlit

Streamlit terdiri dari berbagai fitur. Walaupun fokus pembahasan ini adalah mengenai visualisasi, akan tetapi elemen dalam aplikasi Streamlit tidak hanya berupa grafik visual. Selain grafik visualisasi, teks dan *widgets* juga penting dalam aplikasi yang akan dibuat.

Text Elements

Text elements mengacu pada elemen dalam aplikasi yang berkaitan dengan teks. Selain grafik visualisasi, sebuah aplikasi juga membutuhkan teks. Adapun beberapa text element yang dapat digunakan dalam Streamlit adalah:

Elemen	Deskripsi	Penulisan dalam Streamlit
Title	Judul	<code>st.title("Title")</code>
Header	<i>Header</i>	<code>st.header("Header")</code>
Subheader	<i>Subheader</i>	<code>st.subheader("Subheader")</code>
Markdown	Teks dengan format <i>markdown</i>	<code>st.markdown("**Bold Text**")</code>
Code	Teks dengan format kode Python	<code>st.code("text", language="Python")</code>
Latex	Teks dengan format Latex	<code>st.latex("Latex")</code>

Penggunaan text element dalam Streamlit dicontohkan dalam kode di bawah ini.

```

1 import streamlit as st
2
3 # set the app's title
4 st.title("Text Elements")
5
6 # header
7 st.header("Header in Streamlit")
8
9 # subheader
10 st.subheader("Subheader in Streamlit")
11
12 # markdown
13 st.markdown("This is **bold** and this is _italic_")

```

```

15 # code block
16 code = '''
17 def add(a, b):
18     print("a+b = ", a+b)
19 '''
20 st.code(code, language='python')
21
22 # latex
23 st.latex('''
24 (a+b)^2 = a^2 + b^2 + 2*a*b
25 ''')

```

Adapun hasil yang diperlihatkan oleh kode di atas adalah sebagai berikut.

Text Elements

Header in Streamlit

Subheader in Streamlit

This is **bold** and this is *italic*

```
def add(a, b):
    print("a+b = ", a+b)
```

$$(a + b)^2 = a^2 + b^2 + 2 * a * b$$

Widgets

Widget dapat diartikan sebagai elemen yang mengatur jalannya suatu fungsi ataupun jalannya suatu komponen yang berisikan perintah tertentu. Dalam Streamlit, widget digunakan untuk mempermudah pengguna aplikasi untuk menjalankan dan berinteraksi dengan visualisasi yang diinginkan.

Dalam Streamlit, terdapat banyak *widget* yang dapat digunakan. Dalam modul ini, dijelaskan beberapa *widget* seperti:

- *Button*
- *Toggle*
- *Checkbox*
- *Radio button*
- *Text Input*
- *Slider*

Button

Button digunakan sebagai tombol untuk melakukan suatu interaksi. Ketika *button* diklik, maka *widget* ini akan mengirimkan nilai *True* ke dalam aplikasi. Nilai *True* ini tidak dapat dikembalikan ke *False* kecuali aplikasi di-restart.

```
1 import streamlit as st
2
3 #button
4 if st.button('Click here', help="Click to see the text change"):
5     st.write('Now the button is clicked')
6 else:
7     st.write('The button isn\'t clicked yet')
```

Kode di atas menggambarkan bentuk penggunaan *widget button*. Apabila dijalankan, maka dalam aplikasi akan memiliki tampilan seperti ini.

Click here

The button isn't clicked yet



Click here

Now the button is clicked

Toggle

Apabila ingin menggunakan widget yang dapat memberikan nilai *True* dan dapat dikembalikan ke *False*, maka kita dapat menggunakan *toggle*.

```
1 import streamlit as st
2
3 #toggle
4 toggle_on = st.toggle('Show something')
5
6 if toggle_on:
7     st.write('This is something')
```

Berbeda dengan *button*, bila *toggle* diklik untuk kedua kalinya, maka nilai variabel tempat menyimpan *toggle* akan kembali menjadi *False*.

Show something





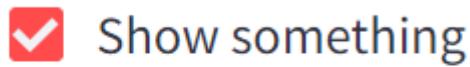
This is something

Checkbox

Checkbox memiliki fungsionalitas yang sama seperti `toggle`, yaitu nilai variabelnya dapat berubah dari `True` ke `False` dan dapat dikembalikan lagi dari `False` ke `True`.

```
1 import streamlit as st
2
3 #toggle
4 check_on = st.checkbox('Show something')
5
6 if check_on:
7     st.write('This is something')
```

Sama juga seperti `toggle`, checkbox disimpan dalam variabel dan memberikan nilai `True` atau `False` tergantung dari apakah `checkbox`-nya digunakan.



This is something

Baik `checkbox` maupun `toggle` dapat digunakan lebih dari satu kali. Adapun yang perlu diingat dalam penggunaan dua `widget` ini adalah tiap `widget` menyimpan satu nilai variabel Boolean.

Radio Button

Berbeda dengan *checkbox* pada umumnya, *radio button* memaksa pengguna untuk hanya memilih satu dari beberapa pilihan. Sehingga nilai kembalian dari variabel *radio button* bukanlah Boolean (dipilih atau tidak; True atau False), melainkan *string* pilihannya. Radio button baik digunakan sebagai kontrol terhadap pilihan, di mana pengguna harus memilih satu dari sekian pilihan.

```
1 import streamlit as st
2
3 # radio button
4 lang = st.radio(
5     "What's your favorite programming language?", 
6     ('C++', 'Python'))
7
8 if lang == 'C++':
9     st.write('You selected C++.')
10 else:
11     st.write('You selected Python.)
```

Bila kode di atas diimplementasikan, maka akan terlihat hasil seperti di bawah ini. Perlu dicatat bahwa pada kode di atas pilihannya hanya ada dua. Hal ini menyebabkan apabila pengguna tidak memilih pilihan pertama, pasti yang terpilih adalah pilihan kedua; dan juga sebaliknya.

What's your favorite programming language?

C++

Python

You selected C++.

What's your favorite programming language?

C++

Python

You selected Python.

Text Input

Text input merupakan bentuk widget yang menerima bukan pilihan, tetapi teks yang dimasukkan oleh pengguna langsung. *Text input* mengembalikan nilai berupa *string* yang sesuai dengan input dari pengguna.

```
1 import streamlit as st  
2  
3 # text input  
4 name = st.text_input('Full Name', 'Abc Xyz')  
5 st.write('The full name is {}'.format(name))
```

Full Name

Abc Xyz

The full name is Abc Xyz

Bentuk lain dari *text input* adalah *number input*, yang secara khusus hanya menerima angka. Sehingga, nilai kembalian dari *number input* adalah *float*. Implementasinya menggunakan *st.number_input*.

Slider

Slider adalah widget lain yang dapat menerima input berupa penggeser. *Slider* menerima input berupa berapa jauh pergeseran, yang mana penggeseran ini diterjemahkan menjadi angka dalam variabel. Sehingga, nilai kembalian dari *slider* adalah *float* atau *integer*.

```
1 import streamlit as st  
2  
3 # slider  
4 score = st.slider('Please specify your test score',  
5 ... min_value=0, max_value=100, value=10)  
6 st.write("My test score is {}".format(score))
```

Please specify your test score



My test score is 38

Visualization with Plotly Express

Di dalam Streamlit, visualisasi dapat menggunakan library apapun: Matplotlib, Seaborn, maupun Plotly. Namun, karena Streamlit memiliki interaksi yang cukup baik dengan *widget*-nya sehingga visualisasinya menarik jika menggunakan Plotly. Pada modul ini, library yang digunakan adalah Plotly Express.

Dibangun di atas plotly.js, plotly.py adalah *library* pembuatan grafik visualisasi tingkat tinggi. plotly.js hadir dengan lebih dari 30 jenis bagan, termasuk grafik ilmiah, grafik 3D, grafik statistik, peta SVG, grafik keuangan, dan banyak lagi.

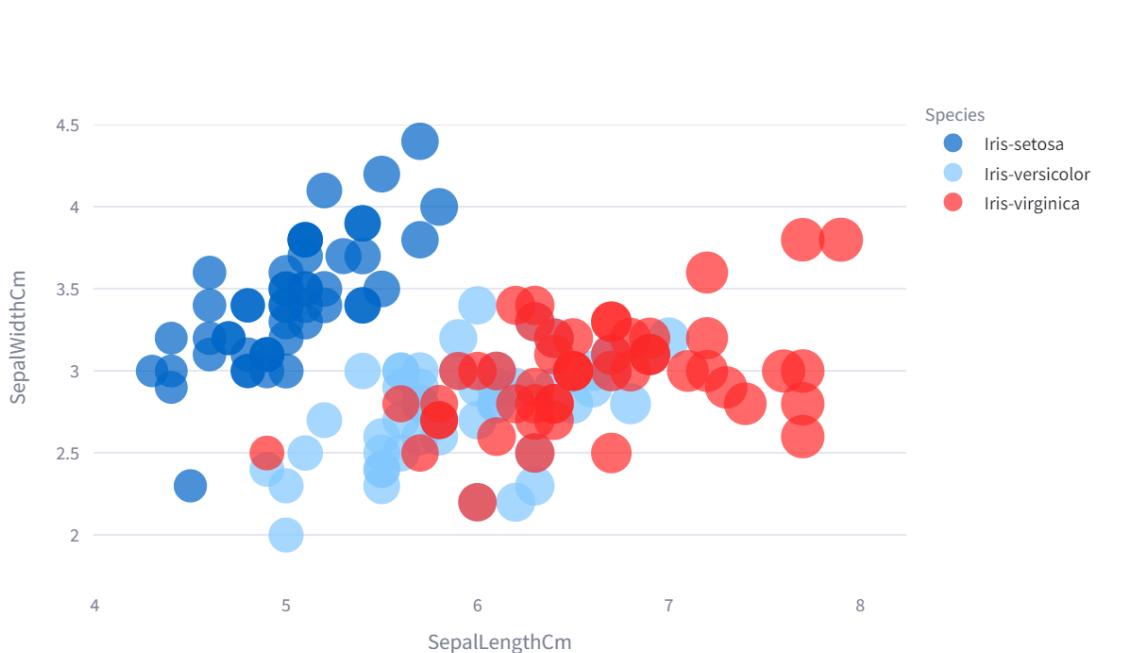
Modul *plotly.express* (biasanya diimpor sebagai *px*) berisi fungsi yang dapat membuat seluruh gambar sekaligus, dan disebut sebagai *Plotly Express* atau *PX*. *Plotly Express* adalah bagian bawaan dari *library* *plotly*, dan merupakan awal yang direkomendasikan untuk membuat grafik visualisasi sederhana. Modul ini menjelaskan bagan-bagan mendasar untuk digunakan dalam Streamlit:

- Scatter
- Line
- Histogram
- Box Plot
- Pie

Scatter Plot

Scatter plot digunakan untuk menggambarkan relasi antara dua variabel kontinyu. Tidak hanya menggambarkan hubungan dua variabel kontinyu, plot dalam *scatter plot* dapat diwarnai dengan variabel diskritnya. Contoh di bawah adalah kode untuk melihat hubungan antara panjang kelopak dengan lebar kelopak, yang dipisahkan berdasarkan spesies dari dataset Iris.

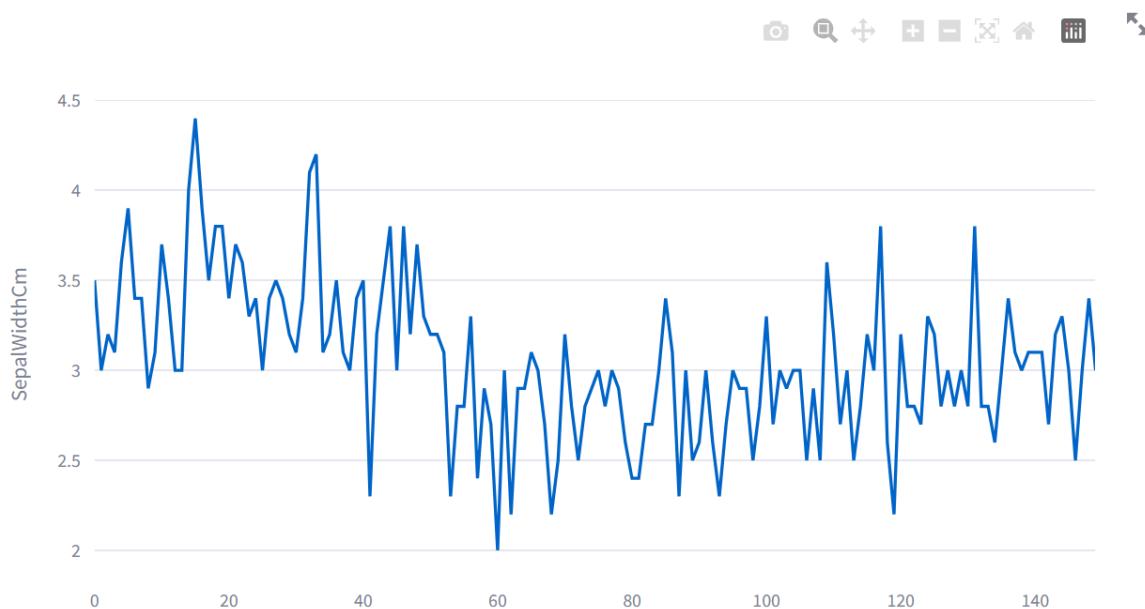
```
1 import pandas as pd
2 import streamlit as st
3 import plotly.express as px
4
5 data = pd.read_csv('iris_data.csv')
6
7 fig = px.scatter(data, x="SepalLengthCm", y="SepalWidthCm", color="Species",
8 ...           size='SepalLengthCm', hover_data=['SepalWidthCm'])
9
10 st.plotly_chart(fig, use_container_width=True)
```



Line Plot

Visualisasi *line plot* umumnya digunakan untuk menggambarkan data yang bersifat seri waktu (*time series*). Oleh karena itu, visualisasi ini akan menggunakan variabel waktu dalam sumbu X-nya dan nilai yang ingin divisualkan dalam variabel Y-nya. Gambar di bawah adalah contoh visualisasi dengan *line plot* dalam Streamlit dengan Plotly Express.

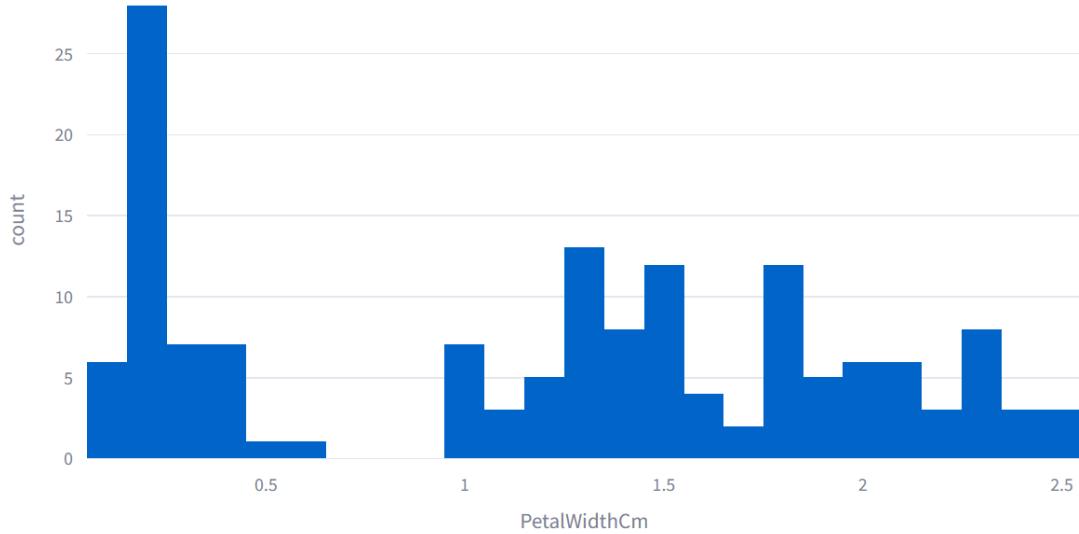
```
7 fig = px.line(data, x=range(len(data)), y="SepalWidthCm",
8 ... ... ... hover_data=['SepalWidthCm'])
9
10 st.plotly_chart(fig, use_container_width=True)
```



Histogram Plot

Plot ini digunakan untuk menggambarkan distribusi data kontinyu. Dalam Plotly Express, px.histogram digunakan sebagai kode untuk memvisualisasikan *histogram plot*. Parameter *nbins* dapat digunakan untuk menentukan jumlah *binning* dalam visualisasi.

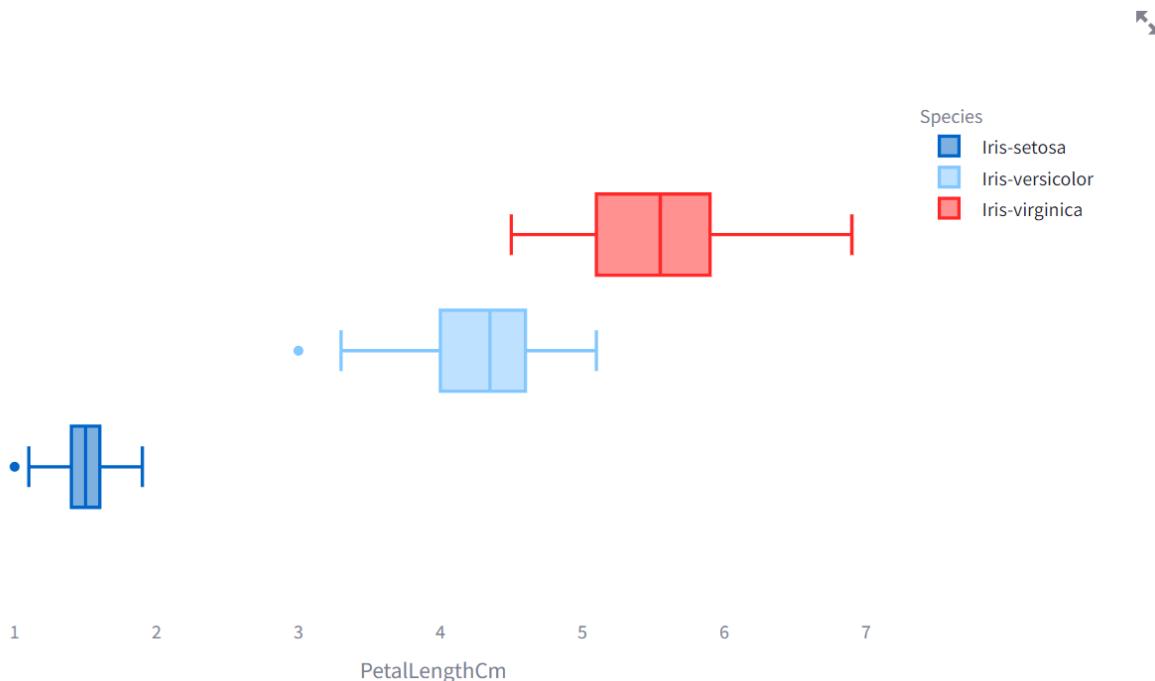
```
7 fig = px.histogram(data, x="PetalWidthCm", nbins=25)
8
9 st.plotly_chart(fig, use_container_width=True)
```



Box Plot

Selain dengan histogram, distribusi data juga dapat digambarkan dengan box plot. Box plot secara khusus digunakan untuk menggambarkan nilai kuartil dan *minmax* dari distribusi data. Oleh sebab itu, data yang divisualisasikan adalah data yang bersifat kontinyu.

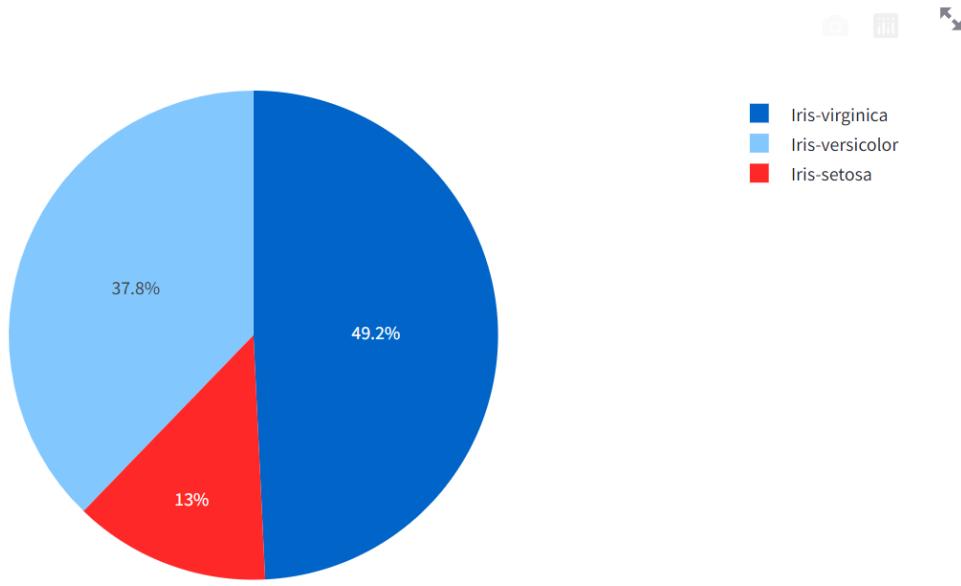
```
7 fig = px.box(data, x="PetalLengthCm", color = 'Species')
8
9 st.plotly_chart(fig, use_container_width=True)
```



Pie Plot

Untuk memvisualisasikan variabel yang bersifat diskrit, *pie plot* dapat dimanfaatkan. Berbeda dengan visualisasi umumnya yang memuat sumbu X dan Y, *pie plot* tidak menggunakan sumbu tersebut. Pie plot sebaiknya digunakan apabila kategori variabel diskrit tersebut kurang dari atau sama dengan tiga.

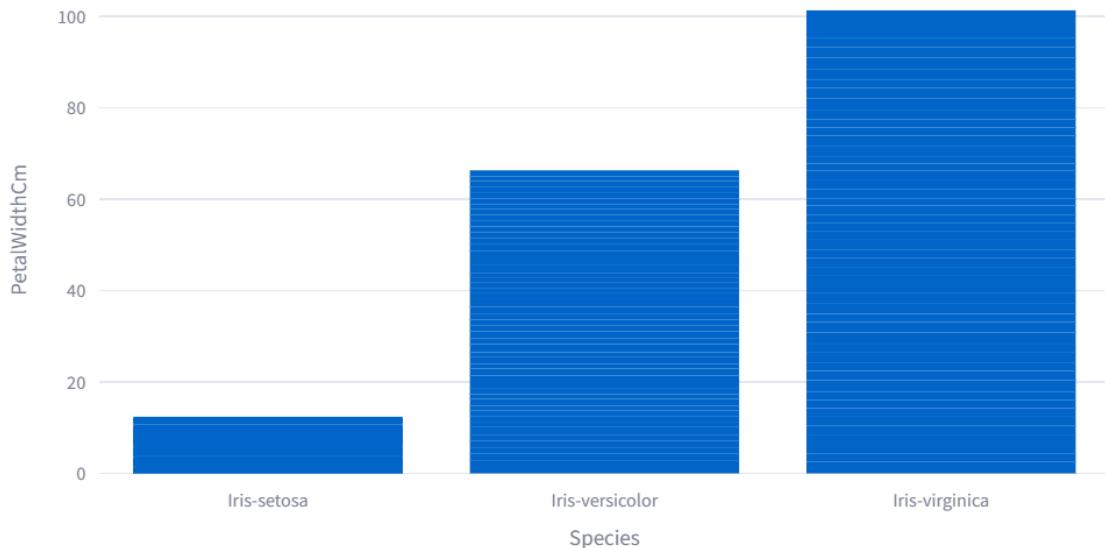
```
7 fig = px.pie(data, values="PetalLengthCm", names = 'Species',)  
8  
9 st.plotly_chart(fig, use_container_width=True)
```



Bar Plot

Selain menggunakan *pie plot*, memvisualisasikan variabel diskrit juga dapat menggunakan *bar plot*. Baik menggunakan *bar plot* ataupun *pie plot* perlu adanya agregasi dalam datanya jika perlu menghitung jumlah anggota ataupun bentuk agregat lainnya.

```
70 fig = px.bar(data, x = 'Species', y = 'PetalWidthCm',)  
71  
72 st.plotly_chart(fig, use_container_width=True)
```



Putting It All Together

Dengan adanya widget untuk mengatur interaksi dan adanya Plotly Express, visualisasi dapat dibuat lebih interaktif dalam Streamlit. Ada beberapa hal yang dapat dilakukan untuk membuat tampilan aplikasi Streamlit lebih menarik.

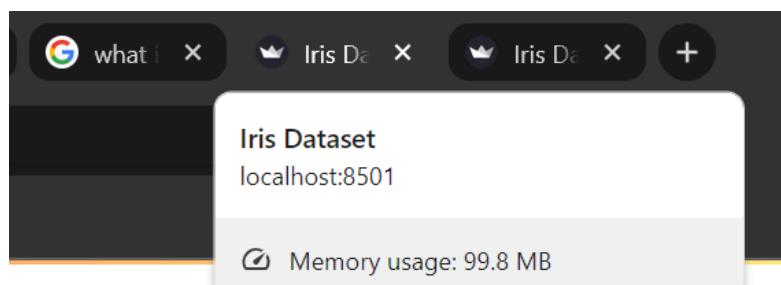
Dalam Streamlit, kita dapat memberikan judul dari aplikasi ketika dimuat dalam browser. Selain itu, kita juga dapat membuat layout aplikasi menjadi lebar. Kedua hal ini dapat dilakukan dengan `st.set_page_config`.

```

7 ▼ st.set_page_config(
8           page_title='Iris Dataset',
9           # layout='wide'
10          )

```

Bila dijalankan, dapat dilihat nama tab dalam browser akan sesuai dengan *page title*.



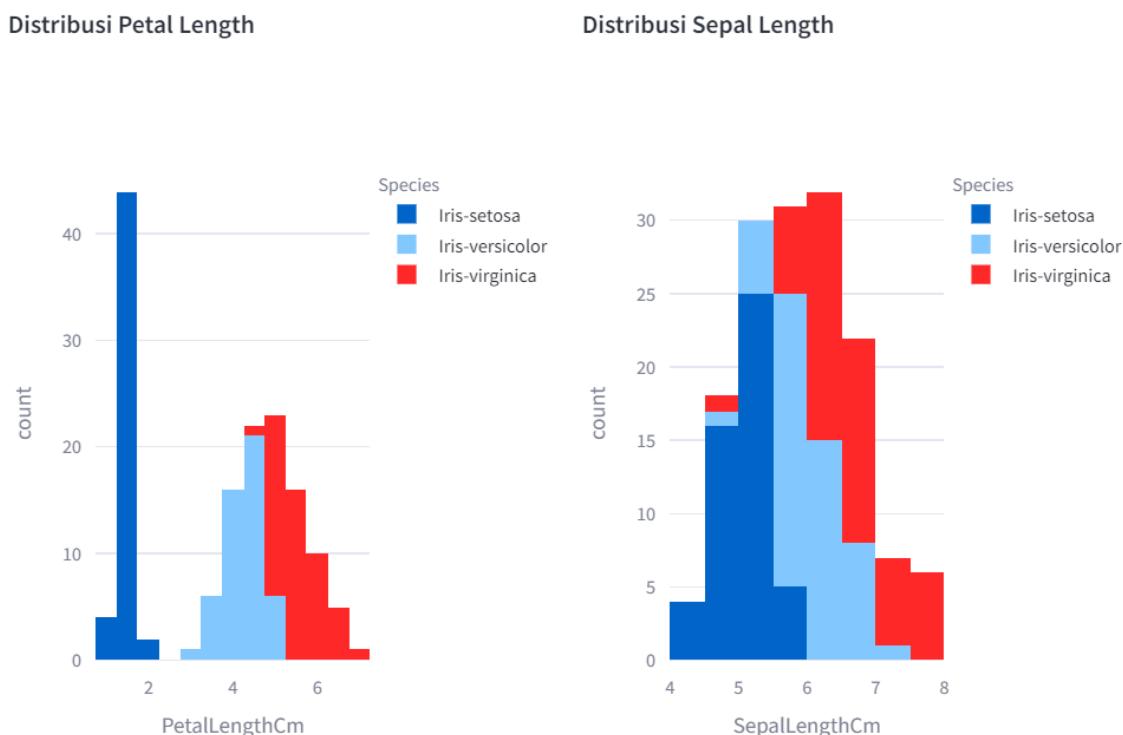
Untuk mengatur layout dari visualisasi ataupun teks, sehingga dua objek tersebut dapat diletakkan bersebelahan (bukan atas-bawah), kita dapat menggunakan fungsi `column` dalam Streamlit. Hal ini dilakukan dengan `st.columns` dan ditunjukkan seperti gambar di bawah.

```

39 col3, col4 = st.columns(2)
40
41 ▶ with col3:
42     st.markdown("##Distribusi Petal Length")
43     fig1 = px.histogram(used_data, x = 'PetalLengthCm', color='Species')
44     st.plotly_chart(fig1, use_container_width=True)
45
46 ▶ with col4:
47     st.markdown("##Distribusi Sepal Length")
48     fig1 = px.histogram(used_data, x = 'SepalLengthCm', color='Species')
49     st.plotly_chart(fig1, use_container_width=True)

```

Ketika menggunakan kolom, tiap kolom didefinisikan dengan `with`. Jika dijalankan, akan terlihat bahwa dua grafik akan ditaruh bersebelahan, bukan bertumpuk.

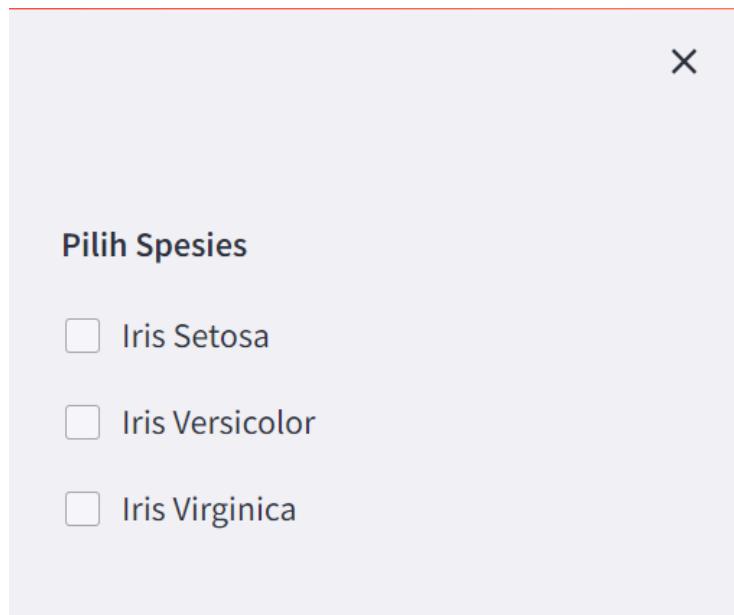


Selain kolom, dalam membuat `layout` kita juga dapat memanfaatkan `sidebar` dengan `st.sidebar`. Sidebar sendiri biasanya tidak berisikan visualisasi, melainkan teks penjelasan ataupun `widget` yang bersifat global.

```

15 ▶ with st.sidebar:
16     st.markdown("##Pilih Spesies")
17     setosa_on = st.checkbox('Iris Setosa')
18     versicolor_on = st.checkbox('Iris Versicolor')
19     virginica_on = st.checkbox('Iris Virginica')

```



Penggunaan teks, layout, widget, dan visualisasi dengan Plotly Express dapat diutak-atik sedemikian rupa untuk menghasilkan aplikasi visualisasi yang menarik dan interaktif.

Membangun Aplikasi AI dengan Streamlit

Streamlit dapat digunakan untuk membangun aplikasi berbasis AI ataupun Machine Learning. Dalam membangun aplikasi ini, terdapat dua tahap utama:

- Membuat model, termasuk dalam *training* dan penyimpanan model
- Membuat aplikasi, termasuk menggunakan model dan membuat tampilan depan

Pada handbook ini, akan dicoba membuat aplikasi untuk memprediksi apakah suatu review pada suatu produk bernuansa positif atau negatif. Hal ini lebih dikenal dengan analisis sentimen.

Membangun Model

Mula-mula, model dibangun layaknya proses dalam machine learning. Dataset yang digunakan adalah dataset review dalam Bahasa Inggris, dengan kelas *review*-nya adalah positif dan negatif. Model yang dibangun menggunakan algoritma Random Forest. Pertama, data di-load dan dipisahkan ke dalam bentuk X (prediktor) dan y (target; kelas).

```
1 import pandas as pd  
2  
1 data = pd.read_excel('product_sentiment.xlsx')  
2 data.head()
```

	review	class
0	So there is no way for me to plug it in here i...	negative
1	Good case, Excellent value.	positive
2	Great for the jawbone.	positive
3	Tied to charger for conversations lasting more...	negative
4	The mic is great.	positive

Kemudian data dipisahkan mana yang menjadi *data training* dan *data testing*.

```

1 from sklearn.model_selection import train_test_split
2
3 X = data['review']
4 y = data['class']
5
6 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

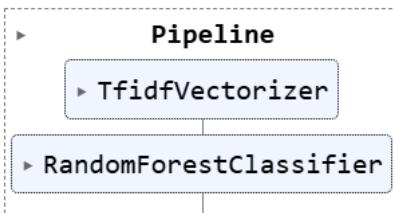
```

Ketika data sudah dipisahkan, maka model dibangun dengan algoritma Random Forest. Adapun teks diubah dalam bentuk vektor dengan TF-IDF. Baik TF-IDF dan Random Forest dibuat dalam satu objek model dalam Sklearn.

```

1 from sklearn.feature_extraction.text import TfidfVectorizer
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn.pipeline import Pipeline
4
5 estimators = [
6     ('vectorizer', TfidfVectorizer()),
7     ('classifier', RandomForestClassifier()),
8 ]
9 model = Pipeline(estimators)
10
11 model.fit(X_train, y_train)

```



Model yang sudah terbentuk diujikan terlebih dahulu.

```

1 from sklearn.metrics import classification_report
2 y_pred = model.predict(X_test)
3 print(classification_report(y_test, y_pred))

```

	precision	recall	f1-score	support
negative	0.68	0.85	0.76	94
positive	0.83	0.65	0.73	106
accuracy			0.74	200
macro avg	0.76	0.75	0.74	200
weighted avg	0.76	0.74	0.74	200

Model yang sudah dirasa cukup akan disimpan. Proses penyimpanan ini dibantu dengan *library* Joblib. Model yang disimpan ini disebut juga dengan Pickle (acar)

```
1 import joblib  
2  
3 joblib.dump(model, 'model.pkl')  
  
['model.pkl']
```

Maka akan muncul file dengan nama Pickle pada direktori kode.



Membangun Aplikasi

Ketika model sudah selesai disimpan dalam format Pickle, maka aplikasi Streamlit tinggal membaca file model dalam bentuk Pickle tersebut. Mula-mula kita ambil *library* yang digunakan.

```
1 import streamlit as st  
2 import plotly.express as px  
3 import joblib  
4 import pandas as pd
```

Kemudian model yang sudah dibangun dan disimpan di-*load* terlebih dahulu dengan Joblib.

```
6 model = joblib.load('model.pkl')
```

Dengan cara ini, model sudah dapat digunakan langsung dalam aplikasi.

Untuk aplikasi sendiri, pertama diatur dulu judul halaman dan judul aplikasi.

```
8 st.set_page_config(  
9     page_title = 'Review Analyzer',  
10    )  
11  
12 st.title('Review Analyzer')
```

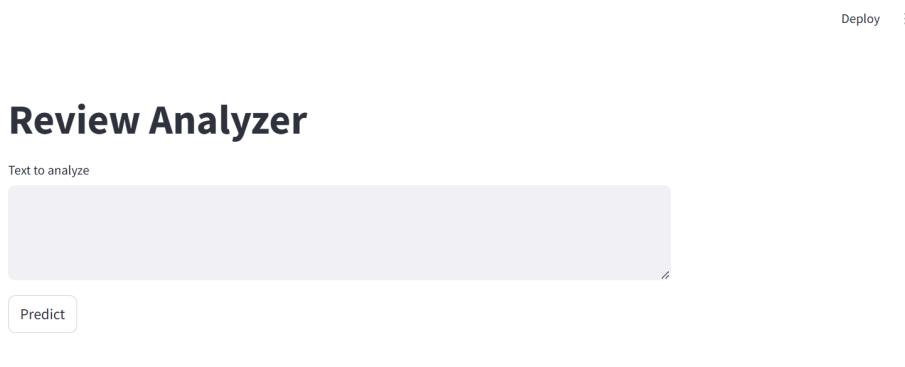
Kemudian, untuk menerima teks yang akan dianalisis sentimennya perlu dibuatkan *field* untuk menerima teks yang panjang.

```
14 text = st.text_area(  
15     "Text to analyze"  
16 )
```

Setelah field untuk teks dibuat, perlu dibuat juga tombol untuk memproses.

```
17 st.button("Predict")
```

Apabila aplikasi ini dijalankan, maka tampilannya sudah seperti ini.



Akan tetapi, ini baru tampilan depan. Walaupun model sudah di-load, antara aplikasi dan model belum ada keterhubungan.

Untuk menggunakan model di dalam aplikasi, maka penggunaan *button* perlu diubah. Pertama, perlu dibuat sebuah *condition* di mana jika teksnya kosong, maka aplikasi tidak akan memproses teksnya. Hal ini dilakukan melalui *button* yang sudah dibuat.

```
18 if st.button("Predict"):  
19     if text.strip() == "":  
20         st.write("The text field is empty")
```

Namun apabila teksnya tidak kosong, maka Streamlit akan memanggil model untuk melakukan prediksi.

```
21 else:  
22     prediction = model.predict([text])[0]  
23     proba = pd.DataFrame()  
24     proba['class'] = ['Negative', 'Positive']  
25     proba['probability'] = model.predict_proba([text])[0]
```

Pada kode di atas, *prediction* pada baris 22 digunakan untuk mengambil hasil sentimen. Kemudian, ditambahkan kondisi output bila sentimennya positif atau negatif.

```
27         if prediction == 'positive':
28             st.write("The sentence is written in positive sentiment")
29         else:
30             st.write("The sentence is written in negative sentiment")
```

Apabila dijalankan, maka aplikasi akan muncul seperti ini.

Deploy :

Review Analyzer

Text to analyze

I do not like this product



Predict

The sentence is written in negative sentiment

Mari buat lebih menarik, dengan menambahkan probabilitas kelas yang disimpan dalam *predict_proba* ke dalam bentuk *pie chart*.

```
53 ▼         fig_pie = px.pie(
54           proba,
55           values='probability',
56           names='class',
57           color='class',
58 ▼           color_discrete_map={
59               'Negative': "red",
60               'Positive': "green",
61           }
62       )
63       st.plotly_chart(fig_pie)
```

Adapun tampilan aplikasinya akan seperti ini.

Review Analyzer

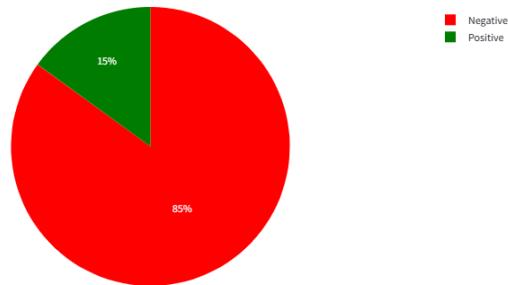
Text to analyze

I do not like this product



Predict

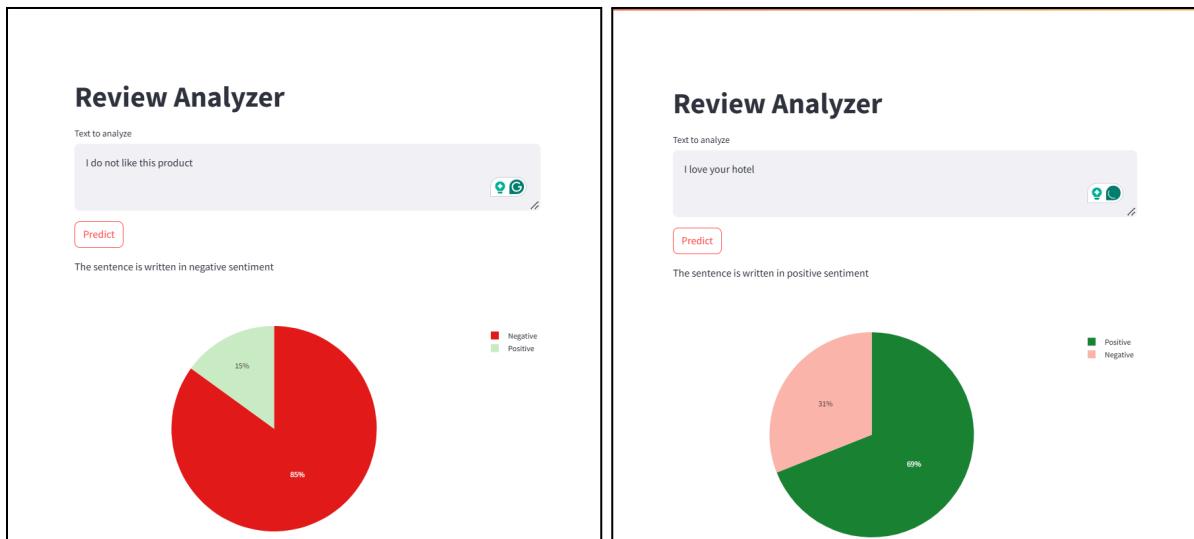
The sentence is written in negative sentiment



Mari buat lebih menarik, dengan cara membuat kelas yang tepat berwarna lebih cerah.

```
28     if prediction == 'positive':
29         st.write("The sentence is written in positive sentiment")
30         fig_pie = px.pie(
31             proba,
32             values='probability',
33             names='class',
34             color='class',
35             color_discrete_map={
36                 'Negative': "rgb(251,180,174)",
37                 'Positive': "#1C8536",
38             }
39         )
40     else:
41         st.write("The sentence is written in negative sentiment")
42         fig_pie = px.pie(
43             proba,
44             values='probability',
45             names='class',
46             color='class',
47             color_discrete_map={
48                 'Negative': "rgb(228,26,28)",
49                 'Positive': "rgb(204,235,197)",
50             }
51         )
```

Dengan mengganti warna untuk tiap kondisi, maka tampilan *pie chart* akan lebih baik.



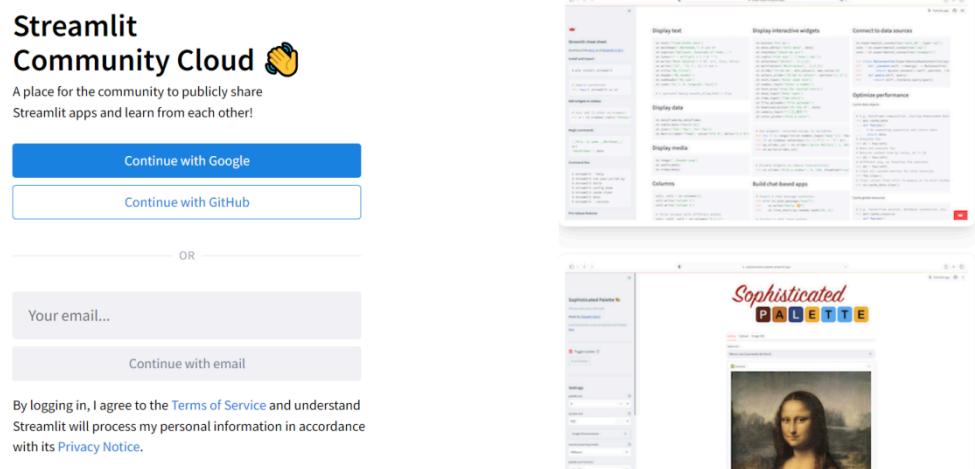
Lakukan modifikasi pada aplikasi ini sehingga terlihat lebih menarik!

Deploy ke Host Streamlit

Aplikasi yang sudah dibuat dapat di-deploy ke hosting Streamlit secara gratis. Hal ini dilakukan melalui share.streamlit.io. Adapun hal yang harus diperhatikan adalah sebagai berikut:

- Memiliki akun Github, dan sudah ada repositori yang memuat kode aplikasi Streamlit kita.
- Repositori harus berisikan aplikasi (dalam format .py) dan *requirements.txt* yang berisikan daftar *library* yang digunakan.
- Kode ataupun file lain dapat ditambahkan ke dalam repositori selama file tersebut memang digunakan dalam aplikasi .py

Apabila hal-hal ini sudah dipenuhi, maka kita dapat langsung menuju *website public hosting* Streamlit. Log-in dapat dilakukan dengan Github (*Continue with Github*).



Kemudian dilanjutkan dengan mengisi data diri.

Set up your account

First Name required
First

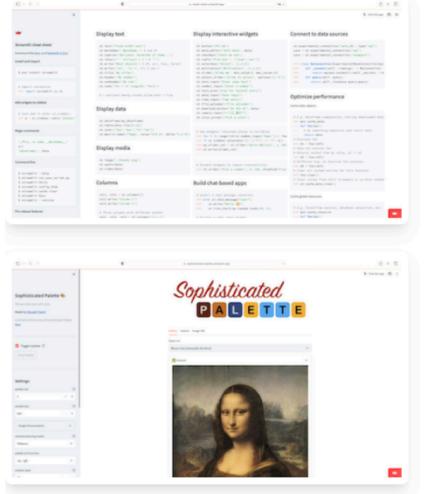
Primary Email required
Email

What's your functional area?
Please select

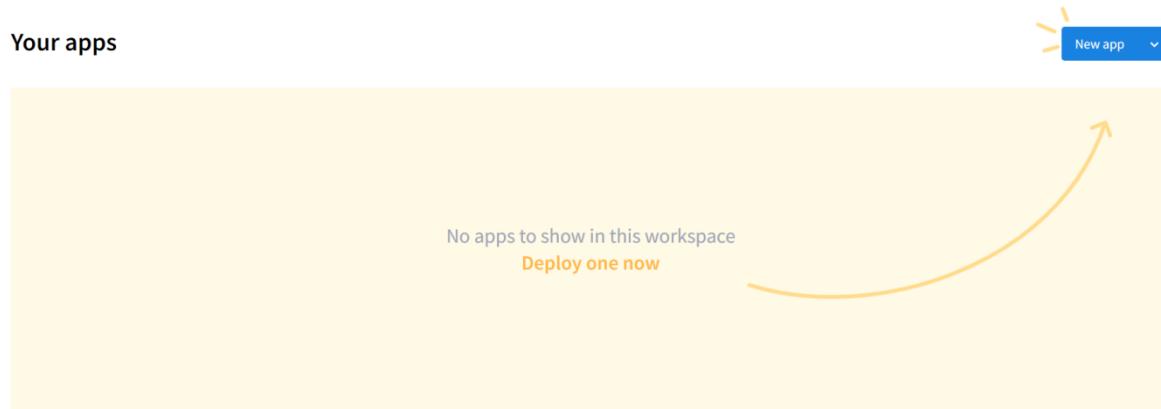
What stage of app development are you at?
Please Select

Want to learn more about using Snowflake with Streamlit?

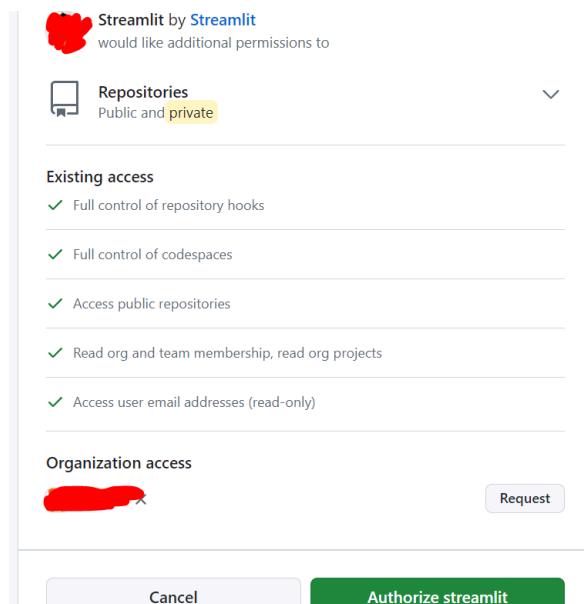
Country or region required
Please Select



Dilanjutkan dengan membuat aplikasi yang sudah masuk ke repositori Github. Tampilan awal akan seperti ini karena belum ada aplikasi yang ada.



Kita perlu memberi ijin untuk Streamlit mengakses Github, sehingga akan muncul tampilan seperti gambar di bawah.



[← Back](#)

Deploy an app

Repository	Paste GitHub URL
prayanatriska/repo	
Branch	
master	
Main file path	
streamlit_app.py	
App URL (Optional)	
.streamlit.app	

[Advanced settings...](#)

Setelah memberi ijin, kita perlu mengisi *form* yang berisikan detail dari repositori dan URL dari aplikasi kita. Apabila tidak diisikan, maka Streamlit akan memberikan URL secara acak. Ketika sudah selesai, maka aplikasi akan diproses seperti di bawah ini, sebelum menjadi aplikasi utuh.

```
[ UTC ] Logs for simpleapp-b5muszf29v9mczkbuewt
[10:21:25] 🚀 Starting up repository: 'simple_strea
[10:21:25] 🚧 Cloning repository...
[10:21:26] 🚧 Cloning into '/mount/src/simple_strea
Warning: Permanently added the ED25519 host key for
[10:21:26] 🚧 Cloned repository!
[10:21:26] 🚧 Pulling code changes from Github...
[10:21:27] 🚧 Processing dependencies...
[10:21:27] 🚧 pip
Using standard pip install.
Collecting plotly==5.10.0 (from -r /mount/src/simple_
  Downloading plotly-5.10.0-py2.py3-none-any.whl.met
Collecting streamlit==1.17.0 (from -r /mount/src/sim
  Downloading streamlit-1.17.0-py2.py3-none-any.whl.
Collecting pandas==1.4.4 (from -r /mount/src/simple_
  Downloading pandas-1.4.4.tar.gz (4.9 MB)
[10:21:27] 🚧 4.9/4.9 M
Installing build dependencies: started
Installing build dependencies: finished with status
Getting requirements to build wheel: started
Getting requirements to build wheel: finished with
Preparing metadata (pyproject.toml): started
Preparing metadata (pyproject.toml): finished with
Collecting tenacity>=6.2.0 (from plotly==5.10.0->_
  Downloading tenacity-8.2.3-py3-none-any.whl.metad
Collecting altair==3.2.0 (from streamlit==1.17.0->_
  Downloading altair-5.3.0-py3-none-any.whl.metadat
[10:21:27] 🚧 master prayanatriska/simple_streamlit/master/hello.py : >
```

Pilih Spesies

- Iris Setosa
- Iris Versicolor
- Iris Virginica

Visualisasi Sederhana dengan Iris

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa



Reference

<https://docs.streamlit.io/develop/api-reference>

<https://plotly.com/python/plotly-express/>

<https://www.askpython.com/python-modules/introduction-to-streamlit>

<https://share.streamlit.io>